REMARKS

Claims 1-16 were presented for examination and were rejected. Reconsideration of this application in view of the following remarks, and allowance of all claims herein, claims 1-16 as written, are hereby respectfully requested.

In her second paragraph, the Examiner rejected claims 1-16 under 35 U.S.C. §102(e) as being anticipated by Chen. Applicant hereby traverses this rejection. All of Applicant's independent claims (claims 1, 13, 14, and 16), and thus all of Applicant's claims, contain two critical limitations that are not suggested by Chen:

1) the code contains instructions causing a macro to be moved to a global environment; and

2) the code contains instructions causing a macro to be copied to a local document.

The Examiner cited column 8, lines 10-22 of Chen for the first proposition (a macro is moved to a global environment). This passage of Chen does not suggest a macro being moved anywhere. The portion of Chen most relevant to Applicant's first critical limitation is column 13 line 50 through column 14 line 24, especially column 14 lines 8-15.

In construing Chen, one should keep in mind that Chen's "template file" is part of what Applicant refers to as "global

environment 13"; and that Chen's "application data file" is analogous to Applicant's "local document 11". Chen further states that a template file can have a .dot extension when the application is Microsoft Word, and an application data file can have a .doc extension when the application is Microsoft Word.

The passage at Chen column 13 line 50 through column 14 line 24 states that one way to detect the presence of a macro is to detect the combination of a "macro enablement instruction" and a "macro reproduction instruction". Chen defines a "macro enablement instruction" as one that formats a file to indicate that the file includes a macro for execution. Column 13 lines 52-54. This does not suggest the "move" nor the "global environment" in Applicant's first critical limitation (instructions causing a macro to be moved to a global environment). Chen states at column 14 lines 8-13 that the instruction "FileSaveAs a$,1" is one example of a macro enablement instruction. Chen defines this instruction to mean "keep an original file and save an additional copy of the file under a different format such as one that indicates that the file can include an embedded macro"(emphasis added). Column 14 lines 9-12. The fact that this instruction requires both a keep and a save implies that it is a copy instruction, not a move instruction as required by the Applicant's first critical limitation. Furthermore, there is no suggestion of a global

environment in this definition. A global environment is also a requirement of Applicant's first critical limitation.

The second prong of Chen's test for the presence of a macro is the presence of a macro reproduction instruction. Column 13 lines 50-52. As an example of a macro reproduction instruction, Chen mentions the instruction "MacroCopy". Column 14 lines 16-20. Chen defines MacroCopy as "copies a macro, and if the macro is infected, all of its harmful instructions, from a source to a destination". Column 14 lines 18-20. This definition discloses the "copy" part of Applicant's second critical limitation (instructions causing a macro to be copied to a local document), but does not suggest the local document requirement in Applicant's second critical limitation.

The above discussion has demonstrated that Applicant's independent claims (claims 1, 13, 14, and 16) are patentable over the cited Chen reference. Applicant's remaining claims are dependent upon his independent claims, and therefore the patentability of these dependent claims follows from the patentability of the independent claims.

Further with respect to claim 5, the SaveAs command recited in claim 5 is for the special case where the code is in Visual Basic. Specification page 12 lines 21-26. Note that when Chen discusses the FileSaveAs command in column 14 lines 8-15, Chen's

application software is WordBasic, not Visual Basic. Column 13 line 64; column 5 lines 37-38.

Similarly, with respect to claim 6, the Copy command recited in claim 6 is for the special case where the code is written in Visual Basic (page 13 lines 15-20), whereas the MacroCopy command that Chen discusses at column 14 lines 16-24 is for WordBasic, not Visual Basic. Column 13 line 64; column 5 lines 37-38.

Further with respect to claim 7, the passage of Chen (column 16 lines 14-40) cited by the Examiner does not suggest the concatenation operator recited in claim 7. Applicant's concatenation operator is described at page 18 lines 3-12 as a symbol, such as an ampersand, that concatenates two strings to assist detection module 17 in finding a location when the writer of malicious code has tried to obscure a location by breaking it into two pieces. The cited passage in Chen, on the other hand, pertains to macro treatment (cleansing).

Further with respect to claim 8, the passage from Chen (column 14 lines 55-64) cited by the Examiner does not suggest Applicant's recitation in claim 8 that "each analyzing step makes substitutions for variable names when the code contains variable names that are proxied" (emphasis added). This recitation is discussed in Applicant's specification at page 18 line 13 through page 19 line 5. The cited passage from Chen suggests neither substitutions nor variable names.

Further with respect to claim 9, the passages of Chen cited by the Examiner (column 14 lines 30-52 and column 16 lines 44-48) do not suggest the recitation in Applicant's claim 9 that "each analyzing step traces the values of parameter variables when the code contains instructions that are invoked by other code" (emphasis added). This recitation is discussed in Applicant's specification at page 19 lines 6 through 22. The cited passages from Chen suggest neither values of parameter variables nor instructions that are invoked by other code.

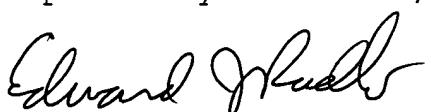
Further with respect to claim 10, the passage of Chen (column 14 lines 55-64) cited by the Examiner does not suggest the recitation of claim 10 that "each analyzing step substitutes object names when the code is written in an object oriented programming language and when the code contains substituted object names" (emphasis added). This recitation is discussed in Applicant's specification at page 19 line 23 through page 20 line 13. The cited passage from Chen suggests neither object names, nor an object oriented programming language, nor substituted object names.

For the above reasons, the Examiner is requested to withdraw her rejection of claims 1-16, and to allow these claims as written.

Applicant believes that this application is now in condition for allowance of all claims herein, claims 1-16 as written, and therefore an early Notice of Allowance is respectfully requested.

If the Examiner disagrees or believes that, for any other reason, direct contact with Applicant's attorney would help advance the prosecution of this case to finality, she is invited to telephone the undersigned at the number given below.

Respectfully submitted,



Edward J. Radlo
Attorney under Rule 34(a)
Reg. No. 26,793

SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box 061080
Wacker Drive Station, Sears Tower
Chicago, IL 60606-1080
Tel.: (415)882-2402

cc: SYMPOL
D. Chi